

DETAILED ACTION

1. Claims 1-53 are pending in this application.
2. This office action is in response to the petition filed on Apr. 8, 2009.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims are 1-4, 6-12, 15-17, 19, 21-24, 27, 38-41, 44-46, and 48-53 are rejected under 35 U.S.C. 102(b) as being anticipated by Smith et al. (US Pat. No. 5,204,960 hereinafter “Smith”).

Per claim 1

Smith discloses

- A method for use in developing a program, comprising compiling at least a portion of a source code program defined by a waypoint during the editing of the source code program (col.2 lines 11-14, lines 20-22 & lines 37-41 e.g. *“Boundaries are established in the source program to define logical blocks within it, each block being termed a function. Each function is further divided into a global region and a local region.... A local region is defined as one in which the consequences of the same given statement are limited to the local region only. The beginning and ending point in each local region in the source file is stored in the .mdt file.... According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.”*)

Per claim 2

the rejection of claim 1 is incorporated and Smith further discloses

- identifying the waypoint in an edited source code during editing of the source code (col.2 lines 20-22 “*The beginning and ending point in each local region in the source file is stored in the .mdt file.*.”)
- compiling the source code up to the identified waypoint before completing the edit of the source code (col.2 lines 37-41 “*According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.*.”)

Per claims 3 and 9

the rejection of claims 1 and 8 are incorporated

Smith discloses in col.2 lines 16-20 “A local region is defined as one in which the consequences of the same given statement are limited to the local region only. The beginning and ending point in each local region in the source file is stored in the .mdt file.” Thus, the local region can be defined by given statements in the source file for

- identifying the waypoint includes one of identifying the waypoint from a static definition and identifying the waypoint from a dynamic definition (col.5 lines 58-61” *if the user has placed a #define statement in a local region, that region must be parsed each time to ensure that changes are recognized.*”, col.5 line 64- col.6 line 1 “*Statements other than a #define statement may also require parsing of the local region each time that region is recompiled, depending on the possibility of global affect of a particular statement. If*

parsing is required, a parsing flag is set when the source file 10 is first compiled for that local region.”)

- Per claims 4 and 10

the rejection of claims 1 and 8 are incorporated and Smith further discloses

- identifying a second waypoint in the source code during editing of the source code; and compiling the source code from the first waypoint to the second waypoint before completing editing of the source code (col.3 lines 32-36 refer to Fig.1 “*Each function 12 is divided into a global region 14 and a local region 16. The global region 14 is first and the local region 16 is second within each function 12, the respective regions alternating throughout the file.*” Like those paragraphs in rejection of claim 1, Boundaries are established in the source code program which is defined by beginning and ending point in each region.)

Per claim 6

the rejection of claim 1 is incorporated and Smith further discloses

- saving the edited source code (col.2 lines 8-11 “*An intermediate file, termed an .mdt file, is generated and stores information about the logical blocks in both the source file and the object file and their relationship to each other.*”)

Per claims 7 and 11

the rejection of claims 1 and 8 are incorporated and Smith further discloses

- compiling the source code from the waypoint to the end of the source code upon completing editing of the source code (col.6 lines 23-33 “*If the local region has changed, (or if a rude construct is within the local region as explained elsewhere herein), the local region must be recompiled, step 344. After the local region 16 is recompiled, the recompiled text may overflow into other regions of the .obj code. If it does not overflow, a “no” in step 344, the incremental compiler returns to step 326 and continues the incremental compiling of the source file as previously explained. If there is an overflow, “yes” in step 344, a complete recompile of all following functions is required and loop 346-352 is entered.”*”)

Per claim 8

Smith discloses

- A method for use in developing a program, comprising: identifying a waypoint in an edited source code program during editing of the source code program (col.2 lines 11-14 & lines 20-22 e.g. “*Boundaries are established in the source program to define logical blocks within it, each block being termed a function. Each function is further divided into a global region and a local region.... A local region is defined as one in which the consequences of the same given statement are limited to the local region only. The beginning and ending point in each local region in the source file is stored in the .mdt file*’’);
- compiling the source code program up to the identified waypoint before completing editing of the source code program (col.2 lines 37-41 “*According to the invention,*

boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.”).

Per claim 12

Smith discloses

A method for modifying a compiler to engage in rapid compilation, comprising:

- identifying a file reader portion of the compiler (col.3 lines 48-52 “*A global region is defined as a region in which a given statement may have consequences throughout the entire file. A local region is defined as one in which the consequences of the same given statement are limited to the local region only.*” The region is interpreted as portion in the source file.)
- modifying the identified file reader to read a portion of a source code program defined by a waypoint from a standard input (col.3 lines 52-58 Refer to Fig.1 “*Thus, if a user edits the file in a global region 14, the entire file following this global region 14 must be recompiled because the edit has the possibility of affecting the remainder of the file. If a user edits the file in a local region 16, only that local region 16 need be recompiled because the affects of statements in a local region 16 are limited to that same local region 16.*”)

Per claim 15

the rejection of claim 12 is incorporated

reject the same reason as claim 3.

Per claim 16

the rejection of claim 12 is incorporated and Smith further discloses

- the waypoint defines a lower bound of the portion of the source code program (col.2 lines 20-22 “ *The beginning and ending point in each local region in the source file is stored in the .mdt file*” & lines 37-41 “ *According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file.*” Boundaries are established in the source code program which is defined by beginning (i.e. Upper bound) and ending point (i.e. lower bound))

Per claim 17

the rejection of claim 12 is incorporated and Smith further discloses

- the waypoint defines an upper bound of the portion of the source code program (col.2 lines 20-22 “ *The beginning and ending point in each local region in the source file is stored in the .mdt file*” & lines 37-41 “ *According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file.*” Boundaries are established in the source code program which is defined by beginning (i.e. Upper bound) and ending point (i.e. lower bound))

Per claim 19

the rejection of claim 18 is incorporated

reject the same reason as claim 3

Per claim 21

the rejection of claim 18 is incorporated and Smith further discloses

- the upper bound of the portion is defined by the start of the source code program or another waypoint (col.2 lines 20-22 “ *The beginning and ending point in each local region in the source file is stored in the .mdt file*” & lines 37-41 “*According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file.*” Boundaries are established in the source code program which is defined by beginning (i.e. Upper bound) and ending point (i.e. lower bound))

Per claim 22

Smith further discloses

A method for resuming compiler execution of a suspended compilation, comprising:

- triggering the compilation of a portion of a source code program whose upper bound is defined by an identified waypoint (col.2 lines 37-41 “*According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.*”)
- and compiling the portion of the source code program whose upper bound is defined by the identified waypoint (col.2 lines 37-41 “*According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a*

change has been made in a particular local region in the source file, only that region is recompiled.”)

Per claim 23

- See the same rejection in claim 22

Per claim 24

Smith discloses

A method for identifying a command and associating it with a file that is being edited, comprising:

- modifying a file reader of a compiler to read from a standard input; and invoking the compiler to read the file from the modified file reader through the standard input (col.3 lines 52-58 *Thus, if a user edits the file in a global region 14, the entire file following this global region 14 must be recompiled because the edit has the possibility of affecting the remainder of the file. If a user edits the file in a local region 16, only that local region 16 need be recompiled because the affects of statements in a local region 16 are limited to that same local region 16.”*)
- triggering the compilation of a portion of a source code program whose upper bound is defined by an identified waypoint (col.2 lines 37-41 *“According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.”*)

Per claim 27

- See the same rejection in claim 24.

Per claim 38

Smith discloses

A method for managing the output of a compile, comprising:

- compiling at least a portion of a source code program defined by a waypoint during the editing of the source code program in a first phase (col.2 lines 11-14, lines 20-22 & lines 37-41 e.g. *“Boundaries are established in the source program to define logical blocks within it, each block being termed a function. Each function is further divided into a global region and a local region.... A local region is defined as one in which the consequences of the same given statement are limited to the local region only. The beginning and ending point in each local region in the source file is stored in the .mdt file.... According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.”*)
- compiling the remainder of the source code program in a subsequent phase (col.9 lines 30-39 *“If the segment count has changed or a segment exceeds its previous size, a "yes" in step 432, a full recompile of all following regions must be performed. If the segment count has not changed and all segments are within their previous sizes, a "no" in step 432, the incremental compiler patches the local region 160 into the object file and*

returns the next global region, step 410, repeating steps 410 to 416 or 410 to 434 as necessary on the remainder of the file.”).

- notifying a user of any errors that may have occurred during the compilation (col.5 lines 39-44 “*During a first compile, many errors are likely in the source file which the user must correct. For certain types of errors, the incremental compiler stops compiling and stops generating the .obj code immediately upon reaching the error and outputs to the user that an error has been found.*”)

Per claim 39

the rejection of claim 38 is incorporated and Smith further discloses

- the portion comprises a portion of the source code program defined by the start of the source code program and the waypoint (col.2 lines 20-22 “*The beginning and ending point in each local region in the source file is stored in the .mdt file*” & lines 37-41 “*According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file.*”)

Per claim 40

the rejection of claim 38 is incorporated and Smith further discloses

- the portion comprises a portion of the source code program defined by the waypoint and the end of the source code program (col.2 lines 20-22 “*The beginning and ending point in each local region in the source file is stored in the .mdt file*” & lines 37-41 “*According to the invention, boundaries are established in the .obj file corresponding to similar*

boundaries in the source file.” & col.4 lines 32-36 refer to Fig.1 “Each function 12 is divided into a global region 14 and a local region 16. The global region 14 is first and the local region 16 is second within each function 12, the respective regions alternating throughout the file.” Like those paragraphs in rejection of claim 1, Boundaries are established in the source code program which is defined by beginning and ending point in each region.)

Per claim 41

the rejection of claim 38 is incorporated

reject the same reason as claim 3

Per claim 44

Smith discloses

A method for use in developing a program, comprising:

- identifying at least two or more instructions in a file to compile; and compiling the identified instructions while the file is being edited (col.4 lines 16-25 “*The non-operational bytes provide a buffer for additional text to be patched into the .obj file and to provide instructions to the linker when building an new .exe file from a recompiled .obj file that has been incrementally compiled. Comment fields per se in an .obj file are known in the prior art. The purpose of comment fields in the prior art is to provide written clues or instruction to a user trying to trace the function of statements in an .obj*

file or to provide directions to other utilities operating on the .obj file such as linkers, debuggers, etc.”).

Per claim 45

the rejection of claim 44 is incorporated and Smith further discloses

- The instructions are identified at a predetermined line number in the source code program, identifying the instructions at the point of insertion for a text editor, identifying the instructions after a predetermined number of branches as conditionals, identifying the instructions at a predetermined text offset (col.8 lines 54-64 “*Each region 160 has a blank comment field at the end of the region for providing instructions to the linker. If line numbers are added in the source file, the number of line numbers added is written into the first two bytes of that ending blank comment field. The linker (and incremental compiler if necessary) reads these first two bytes and adds their value to all subsequent line numbers source file when building the .obj and .exe file to ensure that references to other line numbers of later local regions match up. Providing the number of lines added to (or deleted from) the source file permits the incremental compiler to avoid compile following local regions.”*

Per claim 46

the rejection of claim 44 is incorporated and Smith further discloses

- identifying at least two more instructions in the file during editing; and compiling the second two or more instruction while the file is being edited (col.4 lines 16-25 “*The non-*

operational bytes provide a buffer for additional text to be patched into the .obj file and to provide instructions to the linker when building an new .exe file from a recompiled .obj file that has been incrementally compiled.)

Per claim 48

the rejection of claim 44 is incorporated and Smith further discloses

- comprising saving the edited file (col.2 lines 8-11 “*An intermediate file, termed an .mdt file, is generated and stores information about the logical blocks in both the source file and the object file and their relationship to each other.”*”)

Per claim 49

the rejection of claim 44 is incorporated and Smith further discloses

- compiling the remainder of the edited file upon completing editing of the file (col.9 lines 30-39 “*If the segment count has changed or a segment exceeds its previous size, a "yes" in step 432, a full recompile of all following regions must be performed.)*

Per claim 50

Smith discloses

A method for compiling a source code program, comprising:

- identifying an upper bound for a portion of the source code program to compile; identifying a lower bound for the portion; and compiling the portion defined by the upper and lower bounds during an editing session on the source code program (col.2 lines 11-

14, lines 20-22 & lines 37-41 e.g. “*Boundaries are established in the source program to define logical blocks within it, each block being termed a function. Each function is further divided into a global region and a local region.... A local region is defined as one in which the consequences of the same given statement are limited to the local region only. The beginning and ending point in each local region in the source file is stored in the .mdt file.... According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.*”).

Per claim 51

the rejection of claim 50 is incorporated

Smith discloses in col.2 lines 16-20 “A local region is defined as one in which the consequences of the same given statement are limited to the local region only. The beginning and ending point in each local region in the source file is stored in the .mdt file.” Thus, the local region can be defined by given statements in the source file for at least one of identifying the upper bound and identifying the lower bound includes

- one of identifying the bound from a static definition (col.5 lines 58-61” *if the user has placed a #define statement in a local region, that region must be parsed each time to ensure that changes are recognized.*”) and identifying the bound from a dynamic definition (col.5 line 64- col.6 line 1 “*Statements other than a #define statement may also require parsing of the local region each time that region is recompiled, depending on the*

possibility of global affect of a particular statement. If parsing is required, a parsing flag is set when the source file 10 is first compiled for that local region.”)

Per claim 52

the rejection of claim 50 is incorporated and Smith further discloses

- identifying a third bound in the edited source code during editing of the source code; and compiling the source code from the lower bound to the third bound before completing editing of the source code ((col.3 lines 32-36 refer to Fig.1 “*Each function 12 is divided into a global region 14 and a local region 16. The global region 14 is first and the local region 16 is second within each function 12, the respective regions alternating throughout the file.*” & col.3 lines 52-58 Refer to Fig.1 “*Thus, if a user edits the file in a global region 14, the entire file following this global region 14 must be recompiled because the edit has the possibility of affecting the remainder of the file. If a user edits the file in a local region 16, only that local region 16 need be recompiled because the affects of statements in a local region 16 are limited to that same local region 16.*” Like those paragraphs in rejection of claim 1, Boundaries are established in the source code program which is defined by beginning and ending point in each region.)

Per claim 53

the rejection of claim 50 is incorporated

- compiling the source code from the lower bound to the end of the source code upon completing editing of the source code (see the rejection of claim 52)

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

6. Claims 5, 18 and 47 are rejected under 35 U.S.C. 103(a) as being unpatentable over Smith, in view of Evans et al. (US Pat. No. 6, 836,884 B1, hereinafter “Evans”).

Per claim 5

the rejection of claim 1 is incorporated

Smith discloses the second waypoint (see the rejection of claim 4) and Evans discloses editing a source code program and compiling this source code from a point to the end of the source code (col.5 lines 50-55 *“The method 2 further comprises compiling the edited intermediate language component using an intermediate language compiler (e.g., such as a JIT comailer) at 14 to create an edited native code component and executing the edited native code component at 16 beginning at the first point, whereafter the method 2 ends at 18.”*)

- Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine teaching of Smith of “the second waypoint is defined in the source file” with the teachings of Evans to include “compiling the source code from a point to the end of the source code” in order to reduce the time needed to resume execution of the edited program, avoid unnecessary compiling where an edited portion of the program and allow a software developer to interactively execute portions of the code,

make revisions or changes, and continue execution without having to restart the program execution from the beginning after each edit. (see col.2 lines 42-52)

Per claim 18

Smith discloses

A method for suspending compiler execution prior to reaching the end of a source code program, comprising:

- identifying a waypoint in the source code program (col.2 lines 20-22 “*The beginning and ending point in each local region in the source file is stored in the .mdt file.*”)
- compiling a portion of the source code program whose lower bound is defined by the identified waypoint (col.2 lines 37-41 “*According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file. If a change has been made in a particular local region in the source file, only that region is recompiled.*” Boundaries are established in the source code program which is defined by beginning (i.e. Upper bound) and ending point (i.e. lower bound))

Smith discloses lower bound is defined in the source code program but does not specifically discloses suspending compilation. However, Evans discloses suspending execution of a program at a point (col.2 lines 31-34 “*While program execution is suspended, the user may modify or edit one or more portions of the source code component, and resume execution of the program at the point where execution was suspended.*”).

- Thus, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine Smith and Evans to teach “suspending compilation of the

source code program once the portion whose lower bound is identified by the waypoint is “compiled” in order to reduce the time needed to resume execution of the edited program, avoid unnecessary compiling where an edited portion of the program and allow a software developer to interactively execute portions of the code, make revisions or changes, and continue execution without having to restart the program execution from the beginning after each edit. (see col.2 lines 42-52)

Per claim 47

the rejection of claim 44 is incorporated

Smith does not explicitly disclose completing editing of the file; and compiling the remainder of the edited file.

Evans discloses

- completing editing of the file; and compiling the remainder of the edited file (col.11 lines 52-62 “*Once the execution of the program is suspended or stopped, the edit and continue component 104 allows the user 112 to edit the source code component (e.g., source code component 120 of FIG. 3), for instance, using the source code editor 116 of the debugger application 110 to create an edited source code component. When the desired edits have been completed, the edit and continue component 104 compiles the edited source code component via the interfaces 106 and 108, using the source compiler 118, in order to create an edited intermediate language component.”)*”)
- Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine teaching of Smith with the teachings of Evans to

include “completing editing of the file; and compiling the remainder of the edited file” in order to reduce the time needed to resume execution of the edited program, avoid unnecessary compiling where an edited portion of the program and allow a software developer to interactively execute portions of the code, make revisions or changes, and continue execution without having to restart the program execution from the beginning after each edit. (see col.2 lines 42-52)

7. Claims 13, 14, 25, 26 and 34-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Smith, in view of Meth (U.S. Pub No. 20020087916 A1).

Per claims 13 and 25

the rejection of claim 12 is incorporated

Smith discloses modifying the identified file reader to read from the standard input (col.3 lines 52-58 Refer to Fig.1 “*Thus, if a user edits the file in a global region 14, the entire file following this global region 14 must be recompiled because the edit has the possibility of affecting the remainder of the file. If a user edits the file in a local region 16, only that local region 16 need be recompiled because the affects of statements in a local region 16 are limited to that same local region 16.*”) but Smith does not disclose file reader to read from an open system call.

However, Meth discloses file reader to read from the standard input includes modifying the identified file reader to read from an open system call ([0039] “*whenever the program opens a file with the open() system call, the Condor user-level checkpoint mechanism intercepts the open() system call and records for itself the name of the file being opened*”).

- Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to modify teaching of Evans with the teachings of Meth to include modifying the identified file reader to read from the standard input includes modifying the identified file reader to read from an open system call in order to use to open and close files whenever the program opens a file with the open system call, the user-level checkpoint mechanism intercepts the open system call and records for itself the name of the file being open (see [0039]).

Per claims 14 and 26

the rejection of claim 13 is incorporated and Meth further discloses

- the identified file reader to read from a UNIX gcc command [0039] “... *via standard UNIX system calls...*” which is implicitly included gcc command).

Per claim 34

Smith discloses

- A method for using a UNIX standard input read mechanism for speculative compilation of a source code program, comprising: identifying a waypoint in an edited source code program during editing of the source code program (col.2 lines 20-22 “*The beginning and ending point in each local region in the source file is stored in the .mdt file.*”)

Smith disclose “invoking a compile of at least a portion of a source code program defined by a waypoint during the editing of the source code program” (col.3 lines 52-58 *Thus, if a user edits the file in a global region 14, the entire file following this global region 14 must be recompiled*

because the edit has the possibility of affecting the remainder of the file. If a user edits the file in a local region 16, only that local region 16 need be recompiled because the affects of statements in a local region 16 are limited to that same local region 16.”)

But Smith does not disclose UNIX input read mechanism.

However Meth discloses open or close files using a UNIX input read mechanism ([0039] “*Information about the file descriptors 30 is obtained by the user-level checkpoint mechanism, which in a known manner intercepts the system calls that are used to open and close files, e.g., open(), close(), and dup(). Thus, whenever the program opens a file with the open() system call, ... ”*”)

- Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine teaching of Smith of “invoking a compile of at least a portion of a source code program defined by a waypoint during the editing of the source code program” with the teachings of Meth to include “a UNIX input read mechanism” in order to use to open and close files whenever the program opens a file with the open system call, the user-level checkpoint mechanism intercepts the open system call and records for itself the name of the file being open (see [0039]).

Per claim 35

the rejection of claim 34 is incorporated and Smith further discloses

- the portion comprises a portion of the source code program defined by the start of the source code program and the waypoint (col.2 lines 20-22 “*The beginning and ending point in each local region in the source file is stored in the .mdt file*” & lines 37-41

“According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file.”)

Per claim 36

the rejection of claim 34 is incorporated and Smith further discloses

- the portion comprises a portion of the source code program defined by the waypoint and the end of the source code program (col.2 lines 20-22 *“The beginning and ending point in each local region in the source file is stored in the .mdt file”* & lines 37-41 *“According to the invention, boundaries are established in the .obj file corresponding to similar boundaries in the source file.”* & col.4 lines 32-36 refer to Fig.1 *“Each function 12 is divided into a global region 14 and a local region 16. The global region 14 is first and the local region 16 is second within each function 12, the respective regions alternating throughout the file.”* Like those paragraphs in rejection of claim 1, Boundaries are established in the source code program which is defined by beginning and ending point in each region.)

Per claim 37

the rejection of claim 34 is incorporated

Smith discloses in col.2 lines 16-20 *“A local region is defined as one in which the consequences of the same given statement are limited to the local region only. The beginning and ending point in each local region in the source file is stored in the .mdt file.”* Thus, the local region can be defined by given statements in the source file for

- identifying the waypoint includes one of identifying the waypoint from a static definition and identifying the waypoint from a dynamic definition (col.5 lines 58-61" *if the user has placed a #define statement in a local region, that region must be parsed each time to ensure that changes are recognized.*.", col.5 line 64- col.6 line 1 "Statements other than a #define statement may also require parsing of the local region each time that region is recompiled, depending on the possibility of global affect of a particular statement. If parsing is required, a parsing flag is set when the source file 10 is first compiled for that local region.")

Allowable Subject Matter

8. Claim 20 is objected to as being dependent upon a rejected claim 18, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.
9. Claim 42 and 43 are objected to as being dependent upon a rejected claim 38, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.
10. Claims 28-33 are allowed.

Response to Arguments

Applicant's arguments filed on Feb. 6, 2009 have been fully considered but they are not persuasive.

- In the remarks, Applicant argues that:

(a) In regard to independent claims 1, 12, 18, 22, 24, 38, 44 and 50 recites a “waypoint” and performing some act defined relative to such a waypoint and each of the dependent claims 2-11, 15-17, 19-21 23, 27, 39-43, 45-49, and 51-53 incorporates this limitation. Applicant respectfully submits those claims are novel.

Examiner's response:

Examiner disagrees.

(a) Applicant's arguments with respect to those claims above have been considered but are moot in view of the new ground(s) of rejection - see Smith, Evan and Meth arts made of record, as applied hereto, when considered alone or in combination.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Junchun Wu whose telephone number is 571-270-1250. The examiner can normally be reached on 8:00-17:00 M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

JW
/Wei Y Zhen/

Supervisory Patent Examiner, Art Unit 2191